Maximum H-colourable subdigraphs and constraint optimization with arbitrary weights

Peter Jonsson [a,*],

[a] *Department of Computer and Information Science, Linköpings Universitet, SE-581 83 Linköping, Sweden, tel: +46 13 282415, fax: +46 13 282499*

Andrei Krokhin [b]

[b] *Department of Computer Science, University of Durham, Science Laboratories, South Road, Durham DH1 3LE, UK*

---

* Corresponding author.
Email addresses: Peter.Jonsson@ida.liu.se (Peter Jonsson), andrei.krokhin@durham.ac.uk (Andrei Krokhin).

Preprint submitted to Elsevier Science *22 January 2007*

Abstract: In the maximum constraint satisfaction problem (Max CSP), one is given a finite collection of positive-weight constraints on overlapping sets of variables, and the goal is to assign values from a given domain to the variables so that the total weight of satisfied constraints is maximized. We consider this problem and its variant Max AW CSP where the weights are allowed to be both positive and negative, and study how the complexity of the problems depends on the allowed constraint types. We prove that Max AW CSP over an arbitrary finite domain exhibits a dichotomy: it is either polynomial-time solvable or **NP**-hard. Our proof builds on two results that may be of independent interest: one is that the problem of finding a maximum H-colourable subdigraph in a given digraph is either **NP**-hard or trivial depending on H, and the other a dichotomy result for Max CSP with a single allowed constraint type.

Keywords: maximum constraint satisfaction problem, digraph H-colouring, complexity, dichotomy

1 Introduction and Related Work

The constraint satisfaction problem (CSP) is a powerful general framework in which a variety of combinatorial problems can be expressed [9]. The aim in a constraint satisfaction problem is to find an assignment of values to the variables subject to specified constraints. This framework is used across a variety of research areas in artificial intelligence (see [14,31]), and in computer science, including algorithmic graph theory [22], combinatorial optimization [19,20], database theory [17,26], learning theory [5,11] and complexity theory [9,10,15]. An instance of a constraint satisfaction problem is a set of *constraints* applied to certain specified subsets of variables, and the question is whether there is an assignment to the variables such that all constraint applications are satisfied. The problem of determining how the complexity of CSP (or of one of its many variants) depends on the set $\mathcal{F}$ of constraint types (i.e., predicates) allowed in instances has been thoroughly studied in the last years. Such parameterized problems are denoted CSP($\mathcal{F}$). The first result of this kind was obtained by Schaefer [29] 25 years ago, where he proved that, for all choices of $\mathcal{F}$, CSP($\mathcal{F}$) is either in **P** or **NP**-complete. Furthermore, he gave six classes of Boolean constraints (or rather Boolean relations, or predicates) such that the problem CSP($\mathcal{F}$) is in **P** if and only if all predicates in $\mathcal{F}$ fall entirely within any of these classes. Similar complete classifications of the complexity of constraint problems have been given by, for instance, Bulatov [4] (CSP($\mathcal{F}$) for domains of size 3), Hell and Nešetřil [21] (graph H-colouring) and Creignou *et al.* [9] (various versions of Boolean CSP).

Many different optimization variants of the CSP problem have been suggested. Arguably the most well-known of them is the Max CSP problem where each constraint is assigned a weight and the objective is to find an assignment that maximizes the total weight of the satisfied constraints. This problem is clearly **NP**-hard in general since the Max Cut problem can be viewed as a Max CSP problem (see Example 1). Previously presented complexity results for optimization versions of constraint satisfaction problems, parameterized by the set of allowed constraint types, have mostly been proved under the assumption that only non-negative weights are allowed (*cf.* [6,7,9,24,27]). In the sequel, we will study such problems as well as optimization problems where we allow arbitrary weights. We begin by defining these problems.

Throughout the article D will denote a finite set with $|D| > 1$. Let $R_D^{(m)}$ denote the set of all m-ary predicates over D, that is, functions from D^m to $\{0, 1\}$, and let $R_D = \bigcup_{m=1}^{\infty} R_D^{(m)}$.

Definition 1.1 *A* constraint *over a set of variables $V = \{x_1, x_2, \ldots, x_n\}$ is an expression of the form $f(\mathbf{x})$ where*

- $f \in R_D^{(m)}$ *is called the* constraint function*; and*
- $\mathbf{x} = (x_{i_1}, \ldots, x_{i_m})$ *is called the* constraint scope*.*

The constraint f is said to be satisfied *on a tuple $\mathbf{a} = (a_{i_1}, \ldots, a_{i_m}) \in D^m$ if $f(\mathbf{a}) = 1$.*

Definition 1.2 *Let $\mathcal{F}$ be a finite subset of R_D. An instance of the problem* CSP($\mathcal{F}$) *is a pair (V, C) where*

- *$V = \{x_1, \ldots, x_n\}$ is a set of variables taking their values from D;*
- *C is a collection of constraints $f_1(\mathbf{x}_1), \ldots, f_q(\mathbf{x}_q)$ over V, where $f_i \in \mathcal{F}$ for all $1 \leq i \leq q$.*

The question is whether there is a function $\varphi : V \to D$ which satisfies all constraints in C.

Definition 1.3 *For a finite $\mathcal{F} \subseteq R_D$, an instance of* weighted MAX CSP($\mathcal{F}$) *is a tuple (V, C, ρ) where V, C are the same as for* CSP($\mathcal{F}$)*, and $\rho : C \to \mathbb{Z}^+$ is a function that assigns a positive integral weight ρ_i to each constraint $f_i(\mathbf{x}_i)$. The goal is to maximize the total weight of satisfied constraints, that is, to maximize the function $f : D^n \to \mathbb{Z}^+$, defined by $f(x_1, \ldots, x_n) = \sum_{i=1}^{q} \rho_i \cdot f_i(\mathbf{x}_i)$. The optimal value of a solution to (V, C, ρ) is denoted by $Opt(V, C, \rho)$.*

In the MAX AW CSP($\mathcal{F}$) *problem, we allow the weights to be arbitrary (that is, not necessarily positive) integers.*

Informally speaking, each constraint specifies a property for the variables in its scope, and the weight of a constraint in an instance of MAX CSP expresses the measure of *desirability* for this property to hold, and one needs to find a solution with maximum overall measure of desirability. The problem MAX AW CSP can then be seen as follows: the positive weights express the measure of desirability for certain properties to hold for the scopes of the constraints, while the constraints with negative weights express how *undesirable* it is for certain collections (scopes) of variables to have the properties described by the corresponding constraints; the goal is, again, to maximize the overall desirability. This is useful in, for example, turning constrained optimization into unconstrained optimization which is a common task in mathematical programming (*cf.* [28]). In brief, the constrained problem is modified so that solutions outside the feasible region are *penalized* by giving them large negative weights, and thereafter the modified problem is solved by using an algorithm for unconstrained optimization. Other ways of using constraints to express preferences, with analysis of complexity, can be found in [3,6].

We will study the complexity of problems MAX CSP($\mathcal{F}$) and MAX AW CSP($\mathcal{F}$). Many problems that have received considerable attention in the literature are subsumed by MAX AW CSP($\mathcal{F}$), and prominent examples are

Max k-Cut, Max DiCut and Max k-Sat.

Example 1 *The* Max k-Cut *problem is the problem of partitioning the set of vertices of a given undirected graph with (positive-)weighted edges into k subsets so as to maximize the total weight of edges with ends being in different subsets. This problem is the same as* Max CSP($\{neq_k\}$)*, where neq_k is the binary disequality ($\neq$) predicate on a k-element set, and it is known to be* **NP***-hard (see Problem GT33 in [1]). To see the correspondence between the two problems, view the vertices of the graph as variables and edges as constraint scopes.* Max 2-Cut *is known as simply* Max Cut*.*

Let f_{dicut} be the binary predicate on $\{0,1\}$ with $f_{dicut}(x,y) = 1 \Leftrightarrow x = 0, y = 1$. Then, Max CSP($\{f_{dicut}\}$) *is essentially the problem* Max DiCut *(see problem ND16 in [1]), which is the problem of partitioning the vertices of a digraph with weighted arcs into two subsets V_0 and V_1 so as to maximize the total weight of arcs going from V_0 to V_1. This problem is known to be* **NP***-hard as well.*

If we consider Max AW CSP($\{neq_2\}$) and Max AW CSP($\{f_{dicut}\}$) instead, we see that they correspond to Max Cut and Max DiCut generalised to arbitrary weights, and such problems have been considered by, for instance, Barahona et al. [2] (who point out several important applications) and Goemans & Williamson [16] (who devise an approximation algorithm).

For the Boolean domain, that is, for $|D| = 2$, the complexity of problems Max CSP($\mathcal{F}$) and Max AW CSP($\mathcal{F}$) has been completely classified by Creignou [8] and Jonsson [23], respectively. In both cases, the results appeared to be dichotomies in the sense every such problem is either **NP**-hard or polynomial-time solvable. In this article, we prove that Max AW CSP($\mathcal{F}$) is either polynomial-time solvable or **NP**-hard for any finite domain D, and we also obtain a similar result for Max CSP($\mathcal{F}$) when $\mathcal{F}$ contains a single predicate (the example above indicates that some of the most important Max CSP($\mathcal{F}$) problems are of this kind). The only two previously published complete classifications of complexity for versions of CSP are the results of Dalmau and Jonsson [12] and Grohe [18] where the parameter is, informally, the way in which variables constrain each other (that is, allowed combinations of constraint scopes) rather than the set of allowed constraint predicates.

Recent research pointed out a strong connection between tractability in Max CSP and the algebraic combinatorial property of supermodularity with respect to a lattice ordering of the domain [7,24,27]. We show that our results have the same dividing line: intractable problems identified in this paper do not have this property, while the tractable cases (trivially) do.

The structure of the article is as follows: Section 2 describes our reduction techniques. Section 3 contains the proof of the main result and it is divided

into two parts. In the first part, we study the problem of finding maximum-size H-colourable subdigraphs in digraphs (which is an interesting problem in itself) – we show that it is either **NP**-hard or trivial depending on H. We also show that MAX CSP($\mathcal{F}$), where $\mathcal{F}$ consists of a single predicate, is either **NP**-hard or trivial. These results are used in the second part to give a complete classification of MAX AW CSP. Section 4 exhibits some connections between the results in Section 3 and (super)modularity.

2 Reduction techniques

We present two reduction techniques in this section. From now on, given a subset $D' \subset D$, we let $u_{D'}$ denote a unary predicate such that $u_{D'}(d) = 1$ if and only if $d \in D'$.

2.1 Strict implementations

The first reduction technique in our **NP**-hardness proofs is based on *strict implementations*, see [9,25] where this notion was defined and used only for the Boolean case. We will give this definition in a different form from that of [9,25], but it can easily be checked to be equivalent to the original one (in the case $|D| = 2$).

Definition 2.1 *Let $Y = \{y_1, \ldots, y_m\}$ and $Z = \{z_1, \ldots, z_n\}$ be two disjoint sets of variables. The variables in Y are called primary and the variables in Z auxiliary. The set Z may be empty. Let $g_1(\mathbf{y}_1), \ldots, g_s(\mathbf{y}_s)$, $s > 0$, be constraints over $Y \cup Z$. If $g(y_1, \ldots, y_m)$ is a predicate such that the equality*

$$g(y_1, \ldots, y_m) = \max_Z \sum_{i=1}^{s} g_i(\mathbf{y}_i) - (\alpha - 1)$$

is satisfied for all $y_1, \ldots, y_m$, and some fixed $\alpha \in \mathbb{Z}^+$, then this equality is said to be a strict α-implementation *of g from $g_1, \ldots, g_s$.*

We use $\alpha - 1$ rather than α in the above equality to ensure that this notion coincides with the original notion of a strict α-implementation for Boolean constraints [9,25]. The idea behind strict implementations is that they allow one to modify instances (by substituting predicates) while keeping control over costs of solutions. For example, assume that we have a constraint $g(u, v)$ in an instance of MAX CSP($\mathcal{F}$), and there is a strict 2-implementation $g(y_1, y_2) + 1 = \max_z(g_1(y_1, z) + g_2(z, y_2))$. Then the constraint $g(u, v)$ can be replaced by two constraints $g_1(u, z), g_2(z, v)$ (where z is a fresh variable), and we know

that every solution of cost c to the old instance can be modified (by choosing an appropriate value for z) to a solution of cost $c+1$ to the new instance.

We say that a collection of predicates $\mathcal{F}$ *strictly implements* a predicate g if, for some $\alpha \in \mathbb{Z}^+$, there exists a strict α-implementation of g using predicates only from $\mathcal{F}$.

Lemma 2.2 *If $\mathcal{F}$ strictly implements a predicate g, and* Max CSP$(\mathcal{F} \cup \{g\})$ *is* **NP***-hard, then* Max CSP$(\mathcal{F})$ *is* **NP***-hard as well.*

Proof: We need to show that Max CSP$(\mathcal{F} \cup \{g\})$ is polynomial-time reducible to Max CSP$(\mathcal{F})$. Let

$$g(y_1, \ldots, y_m) = \max_{Z} \sum_{i=1}^{s} g_i(\mathbf{y}_i) - (\alpha - 1), \tag{1}$$

where $g_i \in \mathcal{F}$ for all i.

Let $\mathcal{I}$ be an instance of Max CSP$(\mathcal{F} \cup \{g\})$ corresponding to maximizing the function

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{q} \rho_i \cdot f_i(\mathbf{x}_i). \tag{2}$$

The idea is to transform it to an instance $\mathcal{I}'$ of Max CSP$(\mathcal{F})$ by replacing every constraint in $\mathcal{I}$ whose constraint predicate is g by its strict implementation, introducing new copies of variables from Z each time.

Assume without loss of generality that $f_1 = \ldots = f_r = g$ and $f_i \in \mathcal{F}$ for $r+1 \leq i \leq q$. The constraint $g(\mathbf{x}_1)$ in (2) can be replaced by the right-hand side of equation (1), changing the variables accordingly. Say, if

$$g(x_1, x_2, x_3) = \max_{z_1, z_2} [g_1(x_1, z_1, z_2) + g_2(x_2, z_2, x_3)] - 1$$

and $\mathbf{x}_1 = (x_1, x_2, x_1)$, then $g(\mathbf{x}_1)$ would be replaced by

$$\max_{z_1^1, z_2^1} [g_1(x_1, z_1^1, z_2^1) + g_2(x_2, z_2^1, x_1)] - 1.$$

If we do the same with every constraint $g(\mathbf{x}_i)$, $1 \leq i \leq r$, replacing the primary variables by the corresponding variables from $\mathbf{x}_i$ and using a new set Z_i of auxiliary variables every time, then we obtain that the goal in $\mathcal{I}$ can be restated as that of maximizing the function

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{r} \rho_i \cdot (\max_{Z_i} \sum_{j=1}^{s} g_j(\mathbf{y}_j^i) - (\alpha - 1)) + \sum_{i=r+1}^{q} \rho_i \cdot f_i(\mathbf{x}_i) =$$

$$= \max_{Z_1 \cup \ldots \cup Z_r} \sum_{i=1}^{r} \rho_i \cdot (\sum_{j=1}^{s} g_j(\mathbf{y}_j^i)) + \sum_{i=r+1}^{q} \rho_i \cdot f_i(\mathbf{x}_i) - (\alpha - 1) \cdot \sum_{i=1}^{r} \rho_i.$$

Clearly, maximizing this function is the same as maximizing the function

$$\sum_{i=1}^{r}\sum_{j=1}^{s}\rho_i \cdot g_j(\mathbf{y}_j^i) + \sum_{i=r+1}^{q}\rho_i \cdot f_i(\mathbf{x}_i).$$

Note that this function corresponds to an instance $\mathcal{I}'$ of MAX CSP($\mathcal{F}$) over the set of variables $\{x_1, \ldots, x_n\} \cup \bigcup_{i=1}^{r} Z_i$. Since this transformation can be performed in polynomial time, the result follows. □

The next lemma is a direct application of strict implementations.

Lemma 2.3 *If $\mathcal{F}$ contains two unary predicates u_S, u_T such that $S \cap T = \varnothing$, then* MAX CSP($\mathcal{F}$) *is polynomial-time equivalent to* MAX CSP($\mathcal{F} \cup \{u_{S\cup T}\}$).

Proof: One direction is trivial. The other direction follows from Lemma 2.2, since $u_{S\cup T}(x) = u_S(x) + u_T(x)$ is a strict 1-implementation of $u_{S\cup T}(x)$. □

2.2 Domain restriction

For a subset $D' \subseteq D$, we denote the restriction of a predicate f to D' by $f|_{D'}$, as usual. Let $\mathcal{F}|_{D'} = \{f|_{D'} \mid f \in \mathcal{F} \text{ and } f|_{D'} \text{ is not identically } 0\}$.

Lemma 2.4 *Suppose that $u_{D'} \in \mathcal{F}$ for some $D' \subseteq D$. If* MAX CSP($\mathcal{F}|_{D'}$) *is* **NP***-hard, then so is* MAX CSP($\mathcal{F}$).

Proof: Let $I = (V, C, \rho)$ be an instance of MAX CSP($\mathcal{F}|_{D'}$) and let $K = 1 + \sum_{c\in C}\rho(c)$. We will transform I into an instance I' of MAX CSP($\mathcal{F}$) in polynomial time, in the following way: the set V stays the same; change every constraint $f_i|_{D'}(\mathbf{x}_i)$ in C to $f_i(\mathbf{x}_i)$, add the constraints $c_i = u_{D'}(x_i), x_i \in V$, to C, and extend ρ so that $\rho(c_i) = K$ for all new constraints c_i.

Clearly, in every optimal solution to I', all variables are assigned values from D'. Hence, an optimal solution to I' has value $Opt(I) + K \cdot |V|$ where $Opt(I)$ is the value of an optimal solution to I. □

3 Main results

This section is divided into two subsections: in the first we classify the complexity of MAX CSP($\{h\}$), and in the second one we deal with MAX AW

CSP($\mathcal{F}$). We say that a predicate is *trivial* if it is identically 0.

3.1 Complexity of Max CSP($\{h\}$)

Let $\mathcal{F} = \{h\}$. Clearly, Max CSP($\{h\}$) can be solved in polynomial time if h is unary. Indeed, in any instance of the problem, there is no interaction between different variables, and hence we only need to choose an optimal value for each individual variable. Assume from now on that h is at least binary. We shall say that h is *irreflexive* if $h(d, \dots, d) = 0$ for all $d \in D$.

If h is binary, then it can be considered as a digraph $H = (V_H, A_H)$ where $V_H = D$ and $(a, b) \in A_H \Leftrightarrow h(a, b) = 1$. Recall that, given a digraph $H = (V_H, A_H)$, a digraph $G = (V_G, A_G)$ is called H*-colourable* if there exist a *homomorphism* from G to H, that is, a mapping $\varphi : V_G \to V_H$ such that $(\varphi(x), \varphi(y)) \in A_H$ whenever $(x, y) \in A_G$. In this case, we write $G \to H$. Note that when h is binary then the problem Max CSP($\{h\}$) can be represented as follows:

Max H-col
Instance: Digraph $G = (V, A)$ with weights $w_a \in \mathbb{Z}^+$, $a \in A$.
Goal: Find a maximum weight H-colourable subdigraph of G, that is, $A' \subseteq A$ with maximum total weight $\sum_{a \in A'} w_a$ such that the digraph $G' = (V, A')$ is H-colourable.

Indeed, consider the vertices of G as variables, and introduce a constraint $h(x, y)$, with weight w_a, for every arc $a = (x, y) \in A_G$. This gives a precise correspondence between the two problems.

Recall that a digraph H is called a *core* if every homomorphism from H into itself is injective (that is, an automorphism). It is well known that every digraph H has a unique (up to isomorphism) subdigraph H' such that $H \to H'$ and H' is a core. In this case, the problems Max H-col and Max H'-col are equivalent, and hence we may without loss of generality assume that H is a core.

Let, for simplicity, $D = \{0, \dots, p-1\}$ and let h be an arbitrary binary predicate on D. If H is a digraph associated with h as described above then we say that h is a core if H is a core.

For any $d \in D$, define subsets d^+ and d^- of D by the rules $a \in d^+ \Leftrightarrow h(d, a) = 1$ and $a \in d^- \Leftrightarrow h(a, d) = 1$. Let $\mathcal{U} = \{u_{d^+}, u_{d^-} \mid d \in D\}$.

Lemma 3.1 *Let h be a core. If* Max CSP($\{h\} \cup \mathcal{U}$) *is* **NP***-hard, then so is* Max CSP($\{h\}$).

Proof: Consider the digraph H associated with h. Let $\mathcal{I} = (V, C, \rho)$ be an instance of MAX CSP($\{h\} \cup \mathcal{U}$) and let $K = 1 + \sum_{c \in C} \rho(c)$. Modify $\mathcal{I}$ to get an instance $\mathcal{I}'$ of MAX CSP($\{h\}$) as follows:

(1) Introduce fresh variables $x'_0, \ldots, x'_{p-1}$, construct the following set of constraints $\{h(x'_i, x'_j) \mid h(i,j) = 1\}$, and add these constraints to C with weight K each.
(2) Replace each constraint of the form $u_{d^+}(x)$ in $\mathcal{I}$ by the constraint $h(x'_d, x)$ without changing the weights. Similarly, replace every constraint $u_{d^-}(x)$ by $h(x, x'_d)$ without changing the weights.

Note that the scopes of constraints introduced in step 1 form a digraph G' (with $V_{G'} = \{x'_0, \ldots, x'_{p-1}\}$) isomorphic to H. Take an optimal solution φ to $\mathcal{I}'$. All constraints introduced in step 1 must be satisfied by φ. Hence, since H is a core, $\varphi|_{V_{G'}}$ is an isomorphism from G' onto H. Define $\pi : V_H \to V_H$ as follows: $\pi(i) = j$ whenever $\varphi(x'_j) = i$. Clearly, π is an automorphism (i.e., a injective endomorphism) of H. Moreover, $\varphi' = \pi\varphi$ is also an optimal solution to $\mathcal{I}'$ which, in addition, satisfies the condition $\varphi'(x'_i) = i$ for all $i \in D$.

The construction in step 2 ensures that every optimal solution ψ (say, with value m) to $\mathcal{I}$ can be extended, by letting $\psi(x'_i) = i$ for all $i \in D$, to a solution to $\mathcal{I}'$ with value $m + K \cdot |A_H|$. Since φ is optimal for $\mathcal{I}'$, it follows that the restriction of φ' onto V is an optimal solution to $\mathcal{I}$. Therefore, $Opt(\mathcal{I}') = Opt(\mathcal{I}) + K \cdot |A_H|$ where $Opt(\mathcal{I})$ and $Opt(\mathcal{I}')$ are values of optimal solutions to $\mathcal{I}$ and $\mathcal{I}'$, respectively. Thus, this is a polynomial-time reduction from MAX CSP($\{h\} \cup \mathcal{U}$) to MAX CSP($\{h\}$). □

We will now prove the classification result for MAX CSP($\{h\}$) where h is binary; the basic idea is to use the predicate h to strictly implement certain unary predicates and then apply the previous lemma.

Lemma 3.2 *Let $h : D^2 \to \{0,1\}$ be a non-trivial predicate. If $h(d,d) = 1$ for some $d \in D$, then* MAX CSP($\{h\}$) *is trivial. Otherwise (that is, if h is irreflexive), it is* **NP***-hard.*

Proof: If $h(d,d) = 1$ for some $d \in D$, then the assignment mapping every variable to d satisfies all constraints in any instance. Hence, MAX CSP($\{h\}$) is trivial. Assume now that h is irreflexive. As explained above in this subsection, we may now assume that h is a core (obviously, the core of an irreflexive digraph is also irreflexive). We will prove the result by induction on $|D|$. If $|D| = 2$, then $h(x,y)$ is one of $neq_2(x,y), f_{dicut}(x,y), f_{dicut}(y,x)$ (see Example 1), so we are done. Assume that $|D| > 2$ and, for all irreflexive non-trivial predicates on smaller domains, the result holds. We consider three cases:

Case 1: There exists a $v \in D$ such that $|v^+| > 1$.
If $h|_{v^+}$ is nontrivial, then MAX CSP($\{h|_{v^+}\}$) is **NP**-hard by the inductive assumption, so the problem MAX CSP(h, v^+) is **NP**-hard by Lemma 2.4, so MAX CSP($\{h\}$) is **NP**-hard by Lemma 3.1. Hence, we assume that $h|_{v^+}$ is trivial and note that $D \neq v^+ \cup \{v\}$ since h is a core.

Arbitrarily choose a vertex $w \in v^+$ and note that $v^+ \cap w^- = \varnothing$ since $h|_{v^+}$ is trivial. If $v^+ \cup w^- \subsetneq D$, then MAX CSP($\{h\}$) is **NP**-hard by the inductive assumption ($h|_{v^+ \cup w^-}(v, w) = 1$) and Lemmas 3.1 and 2.4, arguing as above. If $h|_{w^-}$ is nontrivial, then MAX CSP($\{h\}$) is **NP**-hard by the same argument. Otherwise, $h(x, y) = 1$ only if $x \in w^-$ and $y \in v^+$. Assume there exist $w_1 \in w^-$ and $v_1 \in v^+$ such that $h(w_1, v_1) = 0$. Then, $|v_1^-| < |w^-|$ and $v^+ \cup v_1^- \subsetneq D$. We see that $h|_{v^+ \cup v_1^-}(v, v_1) = 1$ and $v^+ \cap v_1^- = \varnothing$. Then, MAX CSP($\{h|_{v^+ \cup v_1^-}\}$) is **NP**-hard by the inductive assumption and MAX CSP($h, v^+ \cup v_1^-$) is **NP**-hard by Lemma 2.4. Consequently, MAX CSP(h, v^+, v_1^-) is **NP**-hard by Lemma 2.3 and MAX CSP($\{h\}$) is **NP**-hard by Lemma 3.1. Finally, if $h(x, y) = 1$ whenever $x \in w^-$ and $y \in v^+$, then H is bipartite. Since h is a core, we have $|D| = 2$ which is a contradiction.

Case 2: There exists $v \in D$ such that $|v^-| > 1$.
This case is analogous to the previous case.

Case 3: Every $v \in D$ satisfies $|v^+| \leq 1$ and $|v^-| \leq 1$.
Pick any $v, w \in D$ such that $h(v, w) = 1$ and note that, since $\{v\} = w^-$ and $\{w\} = v^+$, predicates $u_{\{v\}}$ and $u_{\{w\}}$ are members of $\mathcal{U}$. By the inductive assumption, MAX CSP($\{h|_{\{v,w\}}\}$) is **NP**-hard. As above, we apply Lemmas 2.3, 3.1, and 2.4 to obtain **NP**-hardness of MAX CSP($\{h\}$). □

Finally, we extend the previous lemma to predicates of arbitrary arity via an inductive argument.

Theorem 3.3 *Let $h \in R_D^{(n)}$, $n \geq 2$. If $h(d, \ldots, d) = 1$ for some $d \in D$, then* MAX CSP($\{h\}$) *is trivial. If h is nontrivial and irreflexive, then* MAX CSP($\{h\}$) *is* **NP***-hard.*

Proof: If $h(d, \ldots, d) = 1$ for some $d \in D$, then the assignment mapping every variable to d satisfies all constraints in any instance. Assume that h is nontrivial and irreflexive and show that MAX CSP($\{h\}$) is **NP**-hard. The proof is by induction on n (the arity of h). The basis when $n = 2$ was proved in Lemma 3.2. Assume that the result holds for $n = k$, $k \geq 2$. We show that it holds for $n = k + 1$. Assume first that there exists $(a_1, \ldots, a_{k+1}) \in D^{k+1}$ such that $h(a_1, \ldots, a_{k+1}) = 1$ and $|\{a_1, \ldots, a_{k+1}\}| \leq k$. We assume without loss of generality that $a_k = a_{k+1}$ and consider the predicate $h'(x_1, \ldots, x_k) = h(x_1, \ldots, x_k, x_k)$. Note that this is a strict 1-implementation

of h', that $h'(d,\ldots,d) = 0$ for all $d \in D$, and that h' is nontrivial since $h'(a_1,\ldots,a_k) = 1$. Consequently, MAX CSP($\{h'\}$) is **NP**-hard by the induction hypothesis, and MAX CSP($\{h\}$) is **NP**-hard by Lemma 2.2.

Assume now that $|\{a_1,\ldots,a_{k+1}\}| = k+1$ whenever $h(a_1,\ldots,a_{k+1}) = 1$. Consider the predicate $h'(x_1,\ldots,x_k) = \max_y h(x_1,\ldots,x_k,y)$, and note that this is a strict 1-implementation of h'. We see that $h'(d,\ldots,d) = 0$ for all $d \in D$ (due to the condition above) and h' is non-trivial since h is non-trivial. We can once again apply the induction hypothesis and draw the conclusion that MAX CSP($\{h'\}$) and MAX CSP($\{h\}$) are **NP**-hard. □

3.2 Complexity of MAX AW CSP($\mathcal{F}$)

Theorem 3.6 contains the classification result for MAX AW CSP($\mathcal{F}$); its proof is based on Lemmas 3.4 and 3.5.

Given a predicate $f : D^k \to \{0,1\}$, we say that a variable x_i, $1 \le i \le k$, is *fictitious* in $f(x_1,\ldots,x_k)$ if

$$f(a_1,\ldots,a_{i-1},a_i,a_{i+1},\ldots,a_n) = f(a_1,\ldots,a_{i-1},a_i',a_{i+1},\ldots,a_n)$$

for all choices of $a_1,\ldots,a_{i-1},a_i,a_i',a_{i+1},\ldots,a_k \in D$, and x_i is called *essential* otherwise. We call an n-ary predicate f *essentially unary* if there is a subset $D' \subseteq D$ and an index $1 \le i \le n$ such that $f(x_1,\ldots,x_n) = u_{D'}(x_i)$ for all $x_1,\ldots,x_n \in D$; in other words, $f(x_1,\ldots,x_n) = 1$ if and only if $x_i \in D'$. Note that a predicate is essentially unary if and only if at most one of its variables is essential.

For a predicate f, let $\overline{f} = 1 - f$.

Lemma 3.4 *Let f be a predicate that is not essentially unary. Then,* MAX CSP($\{f,\overline{f}\}$) *is* **NP***-hard.*

Proof: Note that an argument is fictitious in f if and only if it is such in $\overline{f}$. We may without loss of generality assume that the arity of f is even, say $2k$. Moreover, we can assume that f contains at most one fictitious argument if $k > 1$ and no fictitious arguments if f is binary. To justify these assumptions, note that we can repeatedly maximize f and $\overline{f}$ over any one of their fictitious arguments to strictly 1-implement predicates g and $\overline{g}$, respectively, with less fictitious arguments. We stop this process when there is at most one fictitious variable left and the arity of the obtained predicate is even. Since, by the assumption of the theorem, f initially has at least 2 essential variables, the obtained predicate has the required properties. If initially f is of odd arity and

has no fictitious variables, then we can add one by strict 1-implementation $g(x_1, \ldots, x_{2k}) = f(x_1, \ldots, x_{2k-1})$.

Consider the following two functions $f_i : D^{2k} \to \{0, 1, 2\}$:

$$f_1(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{a} \in D^k} [f(\mathbf{x}, \mathbf{a}) + \overline{f}(\mathbf{y}, \mathbf{a})]$$

$$f_2(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{a} \in D^k} [f(\mathbf{a}, \mathbf{x}) + \overline{f}(\mathbf{a}, \mathbf{y})]$$

We show that at least one of them is a strict 2-implementation of a predicate $F_i : D^{2k} \to \{0, 1\}$, that is, $F_i(\mathbf{x}, \mathbf{y}) = f_i(\mathbf{x}, \mathbf{y}) - 1$ for some $1 \leq i \leq 2$. Assume to the contrary that this does not hold, that is, for $i = 1, 2$, there exist $\mathbf{x}_i, \mathbf{y}_i \in D^k$ such that $f_i(\mathbf{x}_i, \mathbf{y}_i) = 0$. Then,

(1) for all $\mathbf{a} \in D^k$, $f(\mathbf{x}_1, \mathbf{a}) = 0$ and $\overline{f}(\mathbf{y}_1, \mathbf{a}) = 0$; and
(2) for all $\mathbf{a}' \in D^k$, $f(\mathbf{a}', \mathbf{x}_2) = 0$ and $\overline{f}(\mathbf{a}', \mathbf{y}_2) = 0$.

We see that for all $\mathbf{a}'$, $f(\mathbf{a}', \mathbf{x}_2) = 0$ so $\overline{f}(\mathbf{a}', \mathbf{x}_2) = 1$. However, for all $\mathbf{a}$, $\overline{f}(\mathbf{y}_1, \mathbf{a}) = 0$ so by setting $\mathbf{a} = \mathbf{x}_2$ and $\mathbf{a}' = \mathbf{y}_1$ we obtain a contradiction. We can consequently assume that at least one of the implementations above is a strict 2-implementation of a predicate F_i. Assume that F_1 is strictly implemented; the other case is analogous.

Claim 1: F_1 is irreflexive
Assume $F_1(d, \ldots, d) = 1$ for some $d \in D$. This implies that there exists an $\mathbf{a} \in D^k$ such that $f(d, \ldots, d, \mathbf{a}) = 1$ and $\overline{f}(d, \ldots, d, \mathbf{a}) = 1$ which is impossible.

Claim 2: F_1 is nontrivial
Assume to the contrary that $F_1(\mathbf{x}, \mathbf{y}) = 0$ for all $\mathbf{x}, \mathbf{y} \in D^k$. Then, for all $\mathbf{x}, \mathbf{y}, \mathbf{a} \in D^k$, exactly one of $f(\mathbf{x}, \mathbf{a})$ and $\overline{f}(\mathbf{y}, \mathbf{a})$ equals 1. If there exist $\mathbf{s}, \mathbf{t}, \mathbf{u} \in D^k$ such that $f(\mathbf{t}, \mathbf{s}) = 1$ and $f(\mathbf{u}, \mathbf{s}) = 0$, then $f(\mathbf{t}, \mathbf{s}) = \overline{f}(\mathbf{u}, \mathbf{s}) = 1$ which leads to a contradiction and F_1 is nontrivial. Otherwise, for all $\mathbf{s} \in D^k$, it holds that either $f(\mathbf{t}, \mathbf{s}) = 1$ for all $\mathbf{t} \in D^k$, or $f(\mathbf{t}, \mathbf{s}) = 0$ for all $\mathbf{t} \in D^k$. In other words, the first k arguments in f are fictitious. By our assumptions on f, it has at most one fictitious variable, and none at all if it is binary. Thus, we reach a contradiction.

We have thus obtained a nontrivial, irreflexive, and at least binary predicate F_1 via strict implementations, so **NP**-hardness of Max CSP($\{f, \overline{f}\}$) follows from Theorem 3.3 and Lemma 2.2. □

For $\mathcal{F} \subseteq R_D$, let $\tilde{\mathcal{F}} = \{f, \overline{f} \mid f \in \mathcal{F}\}$.

Lemma 3.5 *The problems* Max AW CSP($\mathcal{F}$) *and* Max CSP($\tilde{\mathcal{F}}$) *are polynomial-time equivalent for any* $\mathcal{F} \subseteq R_D$,

Proof: Let $\mathcal{I}$ be an instance of MAX CSP($\tilde{\mathcal{F}}$) corresponding to maximizing the function $\sum_{i=1}^{q} \rho_i \cdot f_i(\mathbf{x}_i)$. For $1 \leq i \leq q$, define f_i' to be f_i if $f_i \in \mathcal{F}$, and $\overline{f}_i$ otherwise. Furthermore, for $1 \leq i \leq q$, let ρ_i' be ρ_i if $f_i \in \mathcal{F}$, and $-\rho_i$ otherwise. Finally, let $K = \sum_{f_i \notin \mathcal{F}} \rho_i$. Now it is not hard to see that

$$\sum_{i=1}^{q} \rho_i' \cdot f_i'(\mathbf{x}_i) = \sum_{f_i \in \mathcal{F}} \rho_i \cdot f_i(\mathbf{x}_i) + \sum_{f_i \notin \mathcal{F}} (-\rho_i)(1 - f_i(\mathbf{x}_i)) = \sum_{i=1}^{q} \rho_i \cdot f_i(\mathbf{x}_i) - K.$$

It is clear that $\mathcal{I}$ is equivalent to the instance $\mathcal{I}'$ of MAX AW CSP($\mathcal{F}$) corresponding to maximizing the function $\sum_{i=1}^{q} \rho_i' \cdot f_i'(\mathbf{x}_i)$.

The other direction is very similar. Let $\mathcal{I}$ be an instance of MAX AW CSP($\mathcal{F}$) corresponding to maximizing the function $\sum_{i=1}^{q} \rho_i \cdot f_i(\mathbf{x}_i)$. For $1 \leq i \leq q$, define f_i' to be f_i if $\rho_i > 0$, and $\overline{f}_i$ otherwise. For $1 \leq i \leq q$, let $\rho_i' = |\rho_i|$. Let $K = \sum_{\rho_i < 0} \rho_i'$. Again, it is not hard to see that

$$\sum_{i=1}^{q} \rho_i \cdot f_i(\mathbf{x}_i) = \sum_{i=1}^{q} \rho_i' \cdot f_i'(\mathbf{x}_i) - K.$$

It is clear that $\mathcal{I}$ is equivalent to the instance $\mathcal{I}'$ of MAX CSP($\tilde{\mathcal{F}}$) corresponding to maximizing the function $\sum_{i=1}^{q} \rho_i' \cdot f_i'(\mathbf{x}_i)$. □

Theorem 3.6 *Let $\mathcal{F} \subseteq R_D$. If every predicate in $\mathcal{F}$ is essentially unary, then the problem* MAX AW CSP($\mathcal{F}$) *is tractable. Otherwise,* MAX AW CSP($\mathcal{F}$) *is* **NP***-hard.*

Proof: If every predicate in $\mathcal{F}$ is essentially unary, then, in any instance, there is no variable that constrains any other variable. So, it is possible to greedily choose the value of each variable such that the weight of satisfied constraints is maximized – this yields an optimal solution to the given instance. This process can obviously be carried out in polynomial time. If $\mathcal{F}$ contains a predicate f which is not essentially unary, then MAX CSP($\{f, \overline{f}\}$) is **NP**-hard by Lemma 3.4 and the result follows from Lemma 3.5. □

4 Connections with (super)modularity

Recent studies of the complexity and approximability of MAX CSP [7,24,27] have employed the algebraic property of *supermodularity on lattices* [30]. In this section we investigate how this property relates to the results given in previous sections.

Recall that a partial order on D is called a *lattice* if every two elements $a, b \in D$ have a greatest common lower bound $a \sqcap b$ (*meet*) and a least common upper bound $a \sqcup b$ (*join*). Every lattice can be considered as an algebra $\mathcal{L} = (D, \sqcap, \sqcup)$ with operations meet and join. For more information about lattices, see [13].

If, for $1 \leq i \leq n$, L_i is a lattice on a set D_i, then the product lattice $L_1 \times \ldots \times L_n$ is defined on $D_1 \times \ldots \times D_n$ by extending the operations component-wise, that is, by setting, for $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$, $\mathbf{a} \sqcap \mathbf{b} = (a_1 \sqcap b_1, \ldots, a_n \sqcap b_n)$ and $\mathbf{a} \sqcup \mathbf{b} = (a_1 \sqcup b_1, \ldots, a_n \sqcup b_n)$. A function $f : D_1 \times \ldots \times D_n \to \mathbb{R}$ is said to be *supermodular* on $L_1 \times \ldots \times L_n$ if

$$f(\mathbf{a}) + f(\mathbf{b}) \leq f(\mathbf{a} \sqcap \mathbf{b}) + f(\mathbf{a} \sqcup \mathbf{b})$$

for all $\mathbf{a}, \mathbf{b}$. A function f is called *submodular* if the reverse inequality holds, and *modular* if it is both super- and submodular (that is, the above inequality is an equality). Modular functions are also sometimes called *valuations* [30].

Given a lattice L on D, let L^n denote the n-th power of L, that is, the product of n copies of L. Since predicates on D are functions $D^n \to \{0, 1\}$, it makes sense to speak about modular, super- and submodular predicates on L. We shall say that a set $\mathcal{F} \subseteq R_D$ is modular, super- or submodular on L if all predicates in $\mathcal{F}$ have the corresponding property.

Recall that a distributive lattice is a lattice that can be represented by subsets of a set, with operations being set-theoretic intersection and union. Furthermore, a diamond is a lattice with the following structure: one element is greater than all other elements, one element is smaller than all others, and all other elements are pairwise incomparable. Diamonds with at least 5 elements are not distributive [13].

It is known that if $\mathcal{F}$ only contains predicates that are supermodular on some lattice $\mathcal{L}$, which is distributive or a diamond, then Max CSP($\mathcal{F}$) is tractable (see [7,27], respectively) and there is evidence that *all* polynomial-time solvable cases of Max CSP can be uniformly described by using the concept of supermodularity, at least when the domains are small [7,24]. Moreover, all known tractable cases of Max CSP($\mathcal{F}$) enjoy this property, while all known hard cases do not.

We will now show that, in all hardness results for Max CSP($\mathcal{F}$) obtained in this paper, the set $\mathcal{F}$ is *not* supermodular on any lattice.

Proposition 4.1 *If $f \in R_D^{(n)}$, $n \geq 2$, is nontrivial and irreflexive, then it is not supermodular on any lattice on D.*

Proof: Let $L = (D, \sqcap, \sqcup)$ be any lattice on D. Let $\mathbf{a} = (a_1, a_2, a_3, \ldots, a_n)$, $f(\mathbf{a}) = 1$, and assume that the number $t_{\mathbf{a}} = |\{a_1, \ldots, a_n\}|$ of distinct entries

in $\mathbf{a}$ is minimal among all tuples satisfying f. Since f is irreflexive, not all a_i's are the same. Assume without loss of generality that $a_1 \neq a_2$. Let $\mathbf{b} = (a_2, a_1, a_3, \ldots, a_n)$. Since the operations $\sqcap$ and $\sqcup$ of any lattice are obviously commutative, we have $a_1 \sqcap a_2 = a_2 \sqcap a_1$ and $a_1 \sqcup a_2 = a_2 \sqcup a_1$. Moreover, we have $a_i \sqcap a_i = a_i \sqcup a_i = a_i$ for all $3 \leq i \leq n$. Hence, $t_{\mathbf{a}\sqcap\mathbf{b}}, t_{\mathbf{a}\sqcup\mathbf{b}} < t_{\mathbf{a}}$, and we have $f(\mathbf{a} \sqcap \mathbf{b}) = f(\mathbf{a} \sqcup \mathbf{b}) = 0$ by the assumption on $\mathbf{a}$.

Thus, $f(\mathbf{a}) + f(\mathbf{b}) \not\leq f(\mathbf{a} \sqcap \mathbf{b}) + f(\mathbf{a} \sqcup \mathbf{b})$. □

We will now show that, for any $\mathcal{F}$, the set $\tilde{\mathcal{F}} = \{f, \overline{f} \mid f \in \mathcal{F}\}$ is not supermodular on any lattice on D unless every f in $\mathcal{F}$ (and hence in $\tilde{\mathcal{F}}$) is essentially unary. We remark that every (essentially) unary predicate is (trivially) supermodular on any totally ordered lattice. Now, it is easy to check from the definitions that a predicate f is supermodular on a lattice if and only if $\overline{f}$ is submodular on it. It follows that if $\tilde{\mathcal{F}}$ is supermodular on some lattice L, then it is modular on L. In the rest of this section we will show that any modular predicate on a lattice is essentially unary.

We will use the following result of Topkis (see Theorem 2.6.4 [30]): A *chain* is a totally ordered lattice. Let X_i, $1 \leq i \leq n$, be chains, and $X = X_1 \times \ldots \times X_n$. A function $f : X \to \mathbb{R}$ is said to be *separable* if there exist unary functions $g_i : X_i \to \mathbb{R}$ such that $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} g_i(x_i)$ for all x_i's.

Theorem 4.2 ([30]) *A real-valued function f on X is modular on X if and only if it is separable.*

Corollary 4.3 *Let X be as above. If f is modular on X and the range of f is $\{0, 1\}$, then f is essentially unary.*

Recall that every finite lattice L has the greatest element 1_L and the least element 0_L. We will denote the elements $(0_L, \ldots, 0_L)$ and $(1_L, \ldots, 1_L)$ of L^n by $\mathbf{0}_L$ and $\mathbf{1}_L$, respectively.

Theorem 4.4 *If L is a lattice on D, then every modular predicate on L is essentially unary.*

Proof: Let f be a modular predicate on L. We may assume that f is n-ary, $n \geq 2$, and takes both values 0 and 1, since otherwise there is nothing to prove.

An element $a \in L$ is said to *cover* another element $a' \in L$, denoted $a' \prec a$, if $a' < a$ and there is no $a'' \in L$ with $a' < a'' < a$.

First we show that there exist two elements, $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$, in L^n such that $f(\mathbf{a}) = 0$, $f(\mathbf{b}) = 1$, and $a_i = b_i$ for all posi-

tions i except one, where one of a_i, b_i covers the other in L.

Assume that $f(\mathbf{0}_L) = 0$. Then, in L^n, there is an unrefinable chain, say $\mathbf{0}_L = \mathbf{u}_1 \prec \mathbf{u}_2 \prec \ldots \prec \mathbf{u}_s$, between $\mathbf{0}_L$ and some element $\mathbf{u}_s$ such that $f(\mathbf{u}_s) = 1$. Clearly, there is some $1 \leq j \leq s-1$ such that $f(\mathbf{u}_j) = 0$ and $f(\mathbf{u}_{j+1}) = 1$. It is easy to see that, since $\mathbf{u}_j \prec \mathbf{u}_{j+1}$ in L^n, all coordinates of $\mathbf{u}_j$ and $\mathbf{u}_{j+1}$, except one, coincide, and in this one component the coordinate of $\mathbf{u}_{j+1}$ covers the coordinate of $\mathbf{u}_j$. So we get the required elements $\mathbf{a}$ and $\mathbf{b}$. If $f(\mathbf{0}_L) = 1$, then the argument is very similar.

Assume without loss of generality that $\mathbf{a}$ and $\mathbf{b}$ differ in the first component, that is, $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (a_1', a_2, \ldots, a_n)$ where $a_1 \prec a_1'$ (the case $a_1' \prec a_1$ is very similar). We will show that f essentially depends only on its first coordinate.

For $2 \leq i \leq n$, let $X_i = \{a_i, 1_L\}$, and let $X_1 = \{a_1, a_1'\}$. It is easy to see that every X_i is a chain. Furthermore, $X = X_1 \times \ldots \times X_n$ is a sublattice of L^n. Clearly, the restriction of f to X is a modular function on X. By Corollary 4.3, the function $f|_X$ is essentially unary. Moreover, by the choice of $\mathbf{a}$ and $\mathbf{b}$, we have that $f|_X = g(x_1)$ for some unary predicate g on X_1 such that $g(a_1) = 0$ and $g(a_1') = 1$. In particular, it follows that $f(a_1, 1_L, \ldots, 1_L) = 0$ and $f(a_1', 1_L, \ldots, 1_L) = 1$.

Assume first that $f(\mathbf{1}_L) = 1$. Let $c_2, \ldots, c_n$ be arbitrary elements from L. For $2 \leq i \leq n$, let $X_i' = \{c_i, 1_L\}$, and let $X_1' = \{a_1, 1_L\}$. Furthermore, let $X' = X_1' \times \ldots \times X_n'$. As above, each X_i' is a chain, and X' is a sublattice of L^n. Let $f' = f|_{X'}$. Clearly, f' is modular on X'. Moreover, since $f(\mathbf{1}_L) = 1$ and $f(a_1, 1_L, \ldots, 1_L) = 0$, Corollary 4.3 implies that $f' = g'(x_1)$ for some g' such that $g'(1_L) = 1$. Hence, $f'(1_L, c_2, \ldots, c_n) = 1$. We infer that $f(1_L, x_2, \ldots, x_n) = 1$ for all $x_2, \ldots, x_n \in D$.

Pick any elements $d_1, d_2, \ldots, d_n \in D$ such that $f(d_1, d_2, \ldots, d_n) = 0$. For $1 \leq i \leq n$, let $X_i'' = \{d_i, 1_L\}$, and let $X'' = X_1'' \times \ldots \times X_n''$. By restricting f to X'' and using Corollary 4.3 together with equalities $f(d_1, d_2, \ldots, d_n) = 0$ and $f(1_L, d_2, \ldots, d_n) = 1$, we infer, as above, that $f(d_1, 1_L, \ldots, 1_L) = 0$.

We now take arbitrary $e_2, \ldots, e_n \in D$ and show that $f(d_1, e_2, \ldots, e_n) = 0$. For $2 \leq i \leq n$, let $X_i''' = \{e_i, 1_L\}$, and let $X_1''' = \{d_1, 1_L\}$. Furthermore, let $X''' = X_1''' \times \ldots \times X_n'''$. By restricting f to X''', we can apply Corollary 4.3 again. From the equalities $f(1_L, e_2, \ldots, e_n) = 1$, $f(1_L, 1_L, \ldots, 1_L) = 1$, and $f(d_1, 1_L, \ldots, 1_L) = 0$, we conclude that $f(d_1, e_2, \ldots, e_n) = 0$.

We have shown that, for any $d_1 \in D$, if $f(d_1, d_2, \ldots, d_n) = 0$ for some $d_2, \ldots, d_n \in D$, then we have $f(d_1, x_2, \ldots, x_n) = 0$ for all $x_2, \ldots, x_n$. Thus, f essentially depends only on its first coordinate.

If $f(\mathbf{1}_L) = 0$, then the argument is similar, simply exchange 0 and 1 throughout, and use a'_1 instead of a_1. □

5 Conclusion

We have proved that Max AW CSP over an arbitrary finite domain is either polynomial-time solvable or **NP**-hard, and that the same holds for Max CSP($\{f\}$) where f is an arbitrary predicate on some finite domain. In order to prove these results, we showed that finding a maximum H-colourable subdigraph in a given digraph is either **NP**-hard or trivial depending on H. We have also pointed out some connections between our work and (super)modularity.

Allowing negative weights appeared to have drastic effect on the complexity of Max CSP, since only essentially trivial cases remained tractable. On the positive side, the obtained results agree with ideas of supermodularity-based direction of research in Max CSP [7,24,27]. We believe that further progress in classifying the complexity of Max CSP will be made along the road of integrating methods from algebraic lattice theory and classical combinatorial optimization, with Max CSP being a point of a new connection between the two research areas.

Acknowledgements

Peter Jonsson is supported by the *Swedish Research Council* (VR) under grants 621–2003–3421 and 2006–4532, and the *Center for Industrial Information Technology* (Ceniit) under grant 04.01. Andrei Krokhin is supported by the UK Epsrc grant EP/C543831/1.

References

[1] G. Ausiello, P. Creszenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer, 1999.

[2] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36:493–513, 1988.

[3] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-based CSPs and Valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4(3):199-240, 1999.

[4] A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006.

[5] A. Bulatov, H. Chen, and V. Dalmau. Learnability of relatively quantified generalized formulas. In Proc. 15th International Conference on Algorithmic Learning Theory (ALT-2004), pp. 365–379, volume 3244 of *LNCS*, 2004.

[6] D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin. The complexity of soft constraint satisfaction. *Artificial Intelligence*, 170(11):983-1016, 2006.

[7] D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin. Supermodular functions and the complexity of Max CSP. *Discrete Applied Mathematics*, 149(1-3):53–72, 2005.

[8] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences*, 51:511–522, 1995.

[9] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. 2001.

[10] V. Dalmau. Constraint satisfaction problems in non-deterministic logarithmic space. In Proc. 29th International Colloquium on Automata, Languages and Programming (ICALP-2002), pp. 414–425, volume 2380 of *LNCS*, 2002.

[11] V. Dalmau and P. Jeavons. Learnability of quantified formulas. *Theoretical Computer Science*, 306(1-3): 485–511, 2003.

[12] V. Dalmau and P. Jonsson. The complexity of counting homomorphisms seen from the other side. *Theoretical Computer Science*, 329(1-3): 315–323, 2004.

[13] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[14] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[15] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1998.

[16] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[17] G. Gottlob, L. Leone, and F. Scarcello. Hypertree decomposition and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, 2002.

[18] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. In Proc. 44th IEEE Symposium on Foundations of Computer Science (FOCS-2003), pp. 552–561, 2003.

[19] J. Håstad. On bounded occurence constraint satisfaction. *Information Processing Letters*, 74:1–6, 2000.

[20] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 2001.

[21] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Ser.B*, 48:92–110, 1990.

[22] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[23] P. Jonsson. Boolean constraint satisfaction: Complexity results for optimization problems with arbitrary weights. *Theoretical Computer Science*, 244(1-2):189–203, 2000.

[24] P. Jonsson, M. Klasson, and A. Krokhin. The approximability of three-valued Max CSP. *SIAM Journal on Computing*, 35(3):1329–1349, 2006.

[25] S. Khanna, M. Sudan, L. Trevisan, and D. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001.

[26] Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000.

[27] A. Krokhin and B. Larose. Maximum constraint satisfaction on diamonds. In Proc. 11th International Conference on Principles and Practice of Constraint Programming (CP-2005), pp. 388–402, volume 3709 of *LNCS*, 2005.

[28] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.

[29] T.J. Schaefer. The complexity of satisfiability problems. In Proc. 10th ACM Symposium on Theory of Computing (STOC-1978), pp. 216–226, 1978.

[30] D. Topkis. *Supermodularity and Complementarity*. Princeton University Press, 1998.

[31] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London, 1993.